AMENDMENT TO THE SPECIFICATION

Please amend the specification by marked up replacement paragraph(s) as follows.

Please amend paragraphs 4-6 as follows:

[04] Most applications, however, cannot be executed in the context of a browser. For such applications, the user has to download and execute the application by hand or use a facility such as ActiveX to do so. From a performance aspect, this approach is undesirable, because the downloading phase imposes a large initialization time. If the application is ~~complied~~ compiled to native machine code, as most applications are, executing a downloaded application on the user's computer system raises security concerns, because the application may include a virus or a software defect that damages the user's computer system. JAVA™ is a browser-based programming language that employs virtual machine code rather than native machine code. Even though with JAVA™ applets, which can be embedded into web pages, security concerns are lessened (but still might exist due to security holes in the JAVA™ virtual machine implementation), the initialization overhead is often unacceptably too high.

[05] As a result, there has been some interest in implementing interactive chart navigation in the rendering agent. The rendering capabilities of browsers, however, are primitive, so most authors of web pages adopt a static display approach in presenting their charts interactively. Basically, these web pages reference a single chart at a time. When the user clicks on the chart to drill down, a new web page, which links to another chart with the drill down information, is requested from a server and transmitted to the browser for rendering. This user interface is slow and not easy to use. Mouse clicks are required to bring up the additional chart view. The web page typically displays only one chart, and changing the chart display generally requires fetching another page, which injects delay into the user's exploration process.

[06] Thus, there is a need for a way to implement interactive chart navigation that ~~avoid~~ avoids the security problems and initialization delays of procedural application logic approaches, while being faster and easier to use than typical browser implementations.

Please amend paragraph 7 as follows:

[07] The present invention ~~addressed~~ addresses these and other needs by providing the rendering agent with a computer-readable medium that is marked up to display at least two charts at the same time and, in response to a cursor operation over one of the charts, to replace the other

chart. The cursor operation need not be a mouse click and can simply be the event of moving the cursor into a specified area of the first chart. In the example of a drill down operation, one of the charts may be the main chart such as a pie chart, and the other chart may be used to show the drill down information for one of the items (e.g. a slice) of the main chart. Thus, when the user operates the cursor over the first chart (e.g. the main chart), the rendering agent automatically replaces the second chart (e.g. the drill down chart). Because the rendering agent provides the chart navigation functions, the security and slow initialization problems encountered in prior procedural applications logic are avoided.

Please amend paragraphs 25-27 as follows:

[25] A dynamic rendering agent is a piece of software that is capable of changing the rendered output in response to user actions without having to load or receive additional markup instructions (although, for performance reasons, the dynamic rendering agent may defer requests for images until the images are ready to be displayed). By contrast, a static rendering agent only changes the rendered output by loading a new page of markup. Thus, to display a new image, a static browser typically requires the user to click on a link for a new page that references the new image. When the user clicks on the link, a new page is loaded and rendered. Alternatively, the static browser could use an animated image, but ~~animates~~ animated images are not interactive because they do not change in response to a user's action.

[26] Typically, the dynamic rendering agents ~~permits~~ permit the markup to include the specification of user events and a script of actions to be performed in response to the specified events. The first Internet browsers to permit scripts include NAVIGATOR™ 2 and EXPLORER™ 3. Whenever a scriptable browser loads a web page, the browser organizes the entities marked up in the web page as a hierarchical document object. The Document Object Model (DOM) is a recent attempt by W3C.org to standardize the hierarchical document into a platform- and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of documents and web pages. DOM and other dynamic markup rendering techniques are described in greater detail in Danny Goodman, *Dynamic HTML: The Definitive Reference* (O'Reilly, 1998).

[27] The use of DOM and other dynamic rendering techniques, however, is still very much in its infancy. Although DOM has been used with scriptable browsers to implement image

rollovers (e.g. ~~an~~ a button that changes color or styling when the user moves a mouse cursor over the button), the power of DOM and browser scripting has not as yet been fully realized. In part, this is due to the newness of DOM and the ~~incompatibles~~ incompatibility between MICROSOFT™ and NETSCAPE™ for all but the simplest of scripts. The present invention stems, however, from the realization that particular combinations of such dynamic rendering techniques can be applied in a completely new direction to implement interactive chart navigation in a manner that provides convenience of use without ~~comprising~~ compromising security or imposing large initialization delays.

Please amend paragraph 28 as follows:

[28] In accordance with one embodiment of the present invention, whose operation is illustrated in FIG. 1, dynamic image replacement techniques are used to implement interactive chart navigation. For example, a browser may display a main chart and a drill down chart simultaneously. In response to a user's cursor action over the main chart, the browser replaces the drill down chart that is currently being displayed with another drill down chart that corresponds to the part of the main chart over which the cursor action occurred. Although embodiments of the present invention are illustrated with respect to a working example involving a main chart and drill down charts, the present invention is not limited to these particular kinds of charts and may be profitably employed with other configurations of charts.

Please amend paragraph 32 as follows:

[32] Referring back to FIG. 2, the second image element 230 specifies the second chart, e.g. the default drill down chart, and includes a name attribute 231 of "barchart" that identifies the second chart in DOM and a source attribute ~~223~~ 233 that references the image of the second chart (in a file called "bars/worldpop0.gif"). The second chart is displayed in FIG. ~~2~~ 3 as a drill down chart ~~303~~ 305, which shows the population of China, reported in millions, from 1950 to 2000. The convenience and compactness of charts is evident in this example, because one can tell at a glance that the population of China generally increased from 500 million in 1950 to about 1250 million in 2000, with a brief leveling that took place in the late 1950s at the time of the Great Leap Forward policy.

Please amend paragraph 35 as follows:

[35] The "onMouseOver" attribute 257 specifies the action that the browser is to take in response to the user's movement of the cursor to enter over the hot area. In this case, the action to take is specified by a short script that reassigns the source attribute of the image referenced in DOM object "document.images['barchart'].src" to reference the image stored in the "bar/worldpop0.gif" file. The image referenced as "barchart" is ~~the~~ none other than the second image, defined by the second image element 230's name attribute 231 as having the "barchart" name. Correspondingly, the reassigned source attribute is the source attribute 233 of the second image element 230.

Please amend paragraphs 37-38 as follows:

[37] A second area element 260 defines the second hot area in the image map. In particular, the shape attribute 261 and the coords attribute 263 (some of whose points are abbreviated from ~~of~~ FIG. 2 with an ellipsis for ease of illustration) specify the geometry of the second hot area, which, in the working example, corresponds to the population of India. The href attribute 265 specifies the destination of a hyperlink in case the user clicks in the hot area, and the onMouseOver attribute 267 specifies that the current image named "barchart" is to be replaced by the image in the file referenced as "bars/worldpop1.gif".

[38] Likewise, a third area element 270 defines the third hot area in the image map. In particular, the shape attribute 271 and the coords attribute 273 (some of whose points are abbreviated) specify the geometry of the third hot area, which corresponds to the population of the United States in the working example. The href attribute 275 specifies the destination of a hyperlink in case the user clicks in the hot area, and the onMouseOver attribute 277 specifies that the current image named "barchart" is to be replaced by the image in the file referenced as ~~"bars/worldpop1.gif"~~ "bars/worldpop2.gif".

Please amend paragraphs 40-41 as follows:

[40] When the rendering agent loads the markup embodied in a computer-readable medium, the rendering agent begins to process the markup, format the embedded information, and output the information, e.g. to be displayed on a computer monitor. At point 109, the rendering agent processes the first image element 220 and the second image 230 of FIG. 2 to display the first and second charts simultaneously. A result of this processing is shown on screen 300 in FIG. 3A, with the pie chart 303 ~~is~~ displayed next to the population drill down chart 305. Also shown on

screen 300 is a mouse cursor 301, which is currently positioned over the China slice of the pie chart 303. Because of that position of the mouse cursor 301, the population drill down chart 303 is for China from 1950 to 2000.

[41] When the user rolls the mouse to move the mouse cursor 301, for example from the China slice to the India slice in the pie chart 303 as shown in FIG. 3B, the browser detects this motion and checks to see if this event occurs over any of hot areas of the first chart (FIG. ~~2~~ 1, point 111). In the working example, the geometries of these hot areas have been defined by the shape and coords attributes of the area elements ~~240,~~ 250, 260, and ~~260~~ 270. If there is no such mouse event, then execution loops back to point 109 where the first and second charts ~~continued~~ continue to be displayed simultaneously.

Please amend paragraph 44 as follows:

[44] Accordingly, a technique has been described for interactive chart navigation, in which a chart is dynamically replaced because of mouse ~~evens~~ events that occur on another chart. Thus, movements of the mouse in the pie chart 303 cause the drill down charts 305, 307, and 309 to be automatically replaced and displayed, without having to make any mouse clicks. Moreover, the scripts use only a ~~simply~~ simple assignment instruction to a DOM object, which is compatible with both the NAVIGATOR™ and the EXPLORER™ browsers. The resulting user interface is more convenient and easier to use, because it does not require mouse clicks, but does not require procedural application logic to be downloaded to the browser's computer system and executed, thereby advantageously avoiding its associated security problems and long initialization times.

Please amend paragraph 46 as follows:

[46] The operation of this embodiment is illustrated with respect to a three chart working example, whose markup is stored in a single web page 500, selective portions of which are illustrated in FIG. 5, subdivided into FIGS. 5A, 5B, and 5C. The ~~screens~~ screen displays of this working example are depicted in FIGS. 6A, 6B, and 6C and contain two active charts. A pie chart 603 is active, such that operation of the cursor 601 into each slice invokes a new population bar chart 605 (and its associated image map) and a new age distribution chart 607. The population bar chart 605 is also active, such that operation of the cursor 601 into a bar invokes a new age distribution chart 607 for the corresponding year. Since there are many possible instances of the active population bar chart 603 (e.g. China, India, and the U.S.A), the

replacement process replaces both the population bar chart 605 and the image map associated with the population bar ~~chat~~ chart 607.

Please amend paragraph 49 as follows:

[49] Referring back to FIG. 5A, a second image element 530 specifies the second chart, e.g. a population drill down chart, and includes a name attribute 531 of "barchart" that identifies the second chart in DOM as "barchart" and a source attribute 533 that references the image of the second chart (in a file called "bars/worldpop0.gif"). The second image element 530 also has a usemap attribute 535 that specifies the image map named "worldpop0". The second chart is displayed in FIG. 6A as population drill down chart 605, which shows the population of China, reported in millions, from ~~1950~~ 1996 to 2000.

Please amend paragraphs 54-56 as follows:

[54] Referring back to FIG. ~~1~~ 4, at point 407, the generated markup is made accessible to a computer system executing a rendering agent and loaded into the rendering agent. When the rendering agent loads the markup embodied in a computer-readable medium, the rendering begins to process the markup, format the embedded information, and output the information, e.g. to be displayed on a computer monitor. At point 409, the rendering agent processes the first image element 520, the second image element 530, and the third image 540 element of FIG. 5A to display the first, second, and third charts simultaneously. A result of this processing is shown on screen 600 in FIG. 6A, with the pie chart 603, the population drill down chart 605, and the age distribution chart 607 being displayed simultaneously. In the initial display, the population chart for China and the Chinese age distribution chart for the year 1996 are displayed. Also shown on screen 600 is a mouse cursor 601.

[55] When the user moves the mouse cursor 601, for example to the U.S.A. slice in the pie chart 603 as shown in FIG. 6B, the browser detects this motion and checks to see if this event occurs over any of hot areas of the first chart (FIG. 4, point ~~111~~ 411). In the working example, the geometries of these hot areas have been defined by the shape and coords attributes of the area elements ~~550,~~ 560, 570, and ~~570~~ 580. If there is no such mouse event, then execution loops back to point 421.

[56] If, on the other hand, there is such a mouse event, then execution proceeds to point 413 where the script code specified for that event is processed. Because the mouse event is a

movement of the mouse cursor 601 over the third hot area, defined by the third area element 570, the action specified in the ~~second~~ third area element ~~260~~ 570 in the onMouseOver attribute 577 is performed. That action comprises a series of script statements. The first statement "document.images['barchart'].src = 'bars/worldpop2.gif'" causes the source attribute 533 of the second image 530 (denominated "barchart" by the name attribute 531) to reference the image stored at "bars/worldpop2.gif". The second statement "document.images['barchart'].useMap = '#worldpop2'" causes the useMap attribute 535 of the second image 530 to use the image map named "worldpop2" (i.e. the image map defined by image map element 590). The third script statement "document.images['agechart'].src = 'age/ageUSA1.gif'" causes the age distribution drill down chart to be replaced in the DOM. These replacements cause the browser at point 415 to display a fourth image 609, showing the population of the U.S.A. from 1996 to 2000, and a fifth image 611, showing the age distribution for 1996 in the U.S.A., simultaneously with the pie chart 603.

Please amend paragraph 58 as follows:

[58] Accordingly, a technique has been described for interactive chart navigation, in which ~~a~~ an image map is dynamically replaced in response to mouse events that occur on another chart. This user interface is more convenient and easier to use but does not require procedural application logic to be downloaded to the browser's computer system and executed, thereby advantageously avoiding its associated security problems and long initialization times.

Please amend paragraphs 64-65 as follows:

[64] The computer system 700 can send messages and receive data, including program code, through the network(s), network link 719, and communication interface 717. In the Internet example, a server (not shown) might transmit requested code belonging to an application program for implementing an embodiment of the present invention through the network 725, local network 721 and communication interface 717. The processor ~~704~~ 703 may execute the transmitted code while being received and/or store the code in storage device ~~59~~ 709, or other non-volatile storage for later execution. In this manner, computer system 700 may obtain application code in the form of a carrier wave.

[65] The term "computer-readable medium" as used herein refers to any medium that participates in providing instructions or data to the processor ~~704~~ 703 for execution. Such a

medium may take many forms, including but not limited to non-volatile media, volatile media, and transmission media. Non-volatile media include, for example, optical or magnetic disks, such as storage device 709. Volatile media include dynamic memory, such as main memory 705. Transmission media include coaxial cables, copper wire, and fiber optics, including the wires that comprise bus 701. Transmission media can also take the form of acoustic, optical, or electromagnetic waves, such as those generated during radio frequency (RF) and infrared (IR) data communications. Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, any other magnetic medium, a CD-ROM, CDRW, DVD, any other optical medium, punch cards, paper tape, optical mark sheets, any other physical medium with patterns of holes or other optically recognizable indicia, a RAM, a PROM, and EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave, or any other medium from which a computer can read.